Inter-Office Memorandum

To	Distributed Computing	Date	October 9, 1979
From	Glenn Krasner	Location	Palo Alto
Subject	Tape Server Protocol (Version 0.1)	Organization	PARC/SSL

XEROX

Filed on: [IVY]<TapeServer>Docs>TapeServerProtocol.memo, .press

The Tape Server Protocol is a means for accessing magnetic tapes via a tape server over an Ethernet. It currently provides only a low-level interface to the tape server; in general allowing record-level manipulation rather than file-level manipulation. It does support all the magtape operations understood to be required by Tenex. The protocol uses the Byte Stream Protocol (BSP) and is somewhat based on the File Transfer Protocol (see AltoSubsystems manual) and the CopyDisk Protocol (see <Pup>CopyDisk.press).

Overview

There are two parties involved in mag tape operations: a *server* which controlls the actual tape drive(s) and which responds to commands, and a *user* which makes command requests from the server.

Initially, the *server* is listening for connection requests at a well-known socket (following Rendezvous/Termination Protocol) until the *user* initiates a connection. Once the connection is established, the *user* sends commands and receives answers from the *server* until the connection is terminated.

It is the *operator*'s responsibility to mount the correct tapes on the correct drives; there is no provision in the Protocol to check this.

Similar to the File Transfer and CopyDisk Protocols, the *server* should be passively responding to commands but never initiating action, and the *user* is responsible for checking to be sure all operations are reasonable.

The Protocol

Having once established a BSP connection with the server, the *user* sends a [Version] command to insure that they are using the same protocol version. Typically the *user* then issues an [OpenDrive] command and proceeds to issue comands for that drive. When through, the typical *user* issues a [CloseDrive] command to free the drive, and terminates the connection.

The tape manipulation commands are outlined below, and include reading and writing records, forward and backward skipping of records and files, rewinding and unloading of drives and error/status handling. As mentioned earlier the commands currently do not include higher-level functions such as write/read file or position at file x.

Commands are generally answered by the server with a [Yes] message, meaning that the operation was performed successfully, also in some cases indicating an ending status, or with a [No] message, meaning that the operation was not performed succcessfully, indicating what may have been the

cause.

The protocol allows one *user* to have only one drive open at a time (*i.e*. one drive per connection). In order for one machine to user more than one drive, it must establish multiple connections with the *server*; in effect becoming more than one *user*.

Messages

Commands and responses (generically called *messages*) in the Tape Server Protocol are sent in blocks similar to those of the CopyDisk Protocol. These blocks consist of a length word (which is in words and includes that word), followed generally by a numeric code word distinguishing the message, usually followed by a block of data (record transfers) or string of text (intended to be printed at the operator's console). Blocks are an integral number of words long.

Below is the list of messages. It is organized in terms of the command sent by the *user* followed by corresponding responses sent by the *server*. Following this list is an example of typical tape service, and following that is a complete list of messages along with their currently assigned numbers.

Command/Response List

Identification/Termination:

Command: [Version] <code> <string>
This command should be the first sent by the *user* after establishing a BSP connection. The code should be a numeric code representing the current version number of the *user*'s TSP. String is an arbitrary string such as "Tape Server Protocol, V0.0".
Response: [Version] <code> <string>
The *server* should respond with this message. It is the *user*'s responsibility to match codes for compatibility. String could be displayed at the operator's consold to indicate good connection.
Drive Control:

Command: [OpenDrive] <drive#> <*user*ID>
This command requests the use of drive drive#, providing *user*ID as some sort of password protection. Currently *user*ID is ignored by the *server*.
Response: [Yes] <code-good open>
Should the drive be free and ok for the *user* to have, the *server* responds with this message. Code-good open is used to distinguish this [Yes] message from the many others. Each user (each connection) may have only one drive open at a time.
Response: [No] <code-bad open, type #> <string>
Should the *server* be unable to open this drive for this *user*, this is the appropriate response. The code indicates the reason that the open was unsuccessful, and the string may be sent to the operator.

Command: [CloseDrive]
This command relinquishes control over the drive. It does not request an unload of the tape.
Response: [Yes] <ok op>
Response: [No] <bad close>

Reading/Writing:

Records to be read/written may be of arbitrary length, as far as the protocol is concerned. However, in the Alto implementation they may be no longer than 32K bytes (16K words).

Command: [ReadRecord]
Requests that the next record on the tape be read.
Response: [HereIsRecord] <status> <length> <data>
This is the record requested by the last [ReadRecord]. Status indicates ending status of the read, such as EOT or EOF or error. Data are the data bytes read. Length is the number of bytes

actually read.

Command: [WriteRecord] <length> <data>
Requests that data be written as the next record of the tape.
Response: [Yes] <code-ending status>
All went well with write.
Response: [No] <code-bad ending status> <string>
Something went wrong with write, code and string describe problems.

Skipping:

Command: [FwdSpaceRecord]
Requests the next record forward to be skipped over.
Response: [Yes] <code-ending status>
Response: [No] <code-bad ending status> <string>

Command: [BackSpaceRecord]
Requests a backward skip of one record.
Response: same as [FwdSpaceRecord].

Command: [ForwardSpaceFile]
Requests skipping one file forward.
Response: same as [FwdSpaceRecord]. No indication of number of records skipped is given.

Command: [BackSpaceFile]
Requests skipping back one file.
Response: same as [FwdSpaceRecord].

Writing Spaces:

Command: [WriteEndOfFile]
Requests an EOF be put on the tape, indicating the end of the current file.
Response: [Yes] <code-ending status>
Response: [No] <code-bad ending status> <string>

Command: [WriteBlankTape] <inches>
Requests that inches of tape be written blank.
Response: [Yes] <code-ending status>
Response: [No] <code-bad ending status> <string>

Rewinding:

The [Yes] response to these commands indicates that the operation has been initiated, not completed. This allows the user to initiate a rewind or unload, but not have to wait for its completion.

Command: [Rewind]
Requests a rewind of the tape.
Response: [Yes] <BOT>
Response: [No] <bad Rewind>

Command: [Unload]
Requests that the tape be rewound and unloaded.
Response: [Yes] <BOT>
Response: [No] <bad Unload command>

Status:

Statuses that may be read or written have not yet been fully established. So far they include write

protection, device error (hung device), data rerror, BOT, EOT which may be read; and write protection and suppress error correction, which may be set. These include the statuses required by Tenex.

Command: [GetStatus]
Requests the current status of the drive.
Response: [HereIsStatus] <status block>
Returns status of drive.

Command: [SetStatus] <status> <new setting>
Requests that this drive status be given this new setting.
Response: [Yes] <code-ending status>
Response: [No] <code-bad ending status>

Command: [SendToOperator] <length in bytes> <string>
Requests that the string be printed on the operator's console
Response: [Yes] <ok>
Response: [No] <bad send>

Command: [GetFromOperator]
Requests that a string typed by the operator be sent to the user.
Response: [HereIsText] <length in bytes> <string>
Response: [No] <bad get>

Example

A operator desires to read file 3 from a tape on drive 2. The operator must mount the tape on the drive, put it on line and start the user program. The user establishes connection with the server and the following conversation occurs:

U: [Version] <1> Alto TSP user 1.00
S: [Version] <1> TSP server 1.00
User verifies that versions match, and proceeds.

U: [OpenDrive] <2> <me>
S: [Yes] <good open>

U: [Rewind]
Rewind drive to get to file 3.

U: [FwdSpaceFile]
S: [Yes] <ok op>

U: [FwdSpaceFile]
S: [Yes] <ok op>

U: [FwdSpaceFile]
S: [Yes] <ok op>
At file 3, now read records.

U: [ReadRecord]
S: [HereIsRecord] <ok op> <length in bytes of record> <data of record>
.
.
.
U: [ReadRecord]
S: [HereIsRecord] <ok op> <length in bytes of record> <data of record>

U: [ReadRecord]
S: [HereIsRecord] <EOF> <length in bytes of record> <data of record>
End of File, last record read.

U: [CloseDrive]
S: [Yes] <ok op>

Connection is terminated; all went well.

Message List

Message	*Mark Byte Value(octal)*
[Yes]	1
[No]	2
[Version]	3
[SendToOperator]	4
[GetFromOperator]	5
[OpenDrive]	6
[CloseDrive]	7
[ReadRecord]	10
[WriteRecord]	11
[FwdSpaceRecord]	12
[BackSpaceRecord]	13
[FwdSpaceFile]	14
[BackSpaceFile]	15
[WriteEndOfFile]	16
[WriteBlankTape]	17
[Rewind]	20
[Unload]	21
[GetStatus]	22
[SetStatus]	23
[HereIsText]	24
[HereIsRecord]	25
[HereIsStatus]	26

Ending Codes (Yes, No response codes)

0	Ok op, operation went well, no problems
1	BOT, last op resulted in Beginning of Tape condition
2	EOT, last op resulted in End of Tape condition
3	EOF, last op resulted in End Of File condition
4	Drive offline error
5	Hardware error
6	Software error
7	Illegal handle
10	Write attempted on write-protected tape

The file TSP.decl includes the BCPL declarations for the TSP messages.